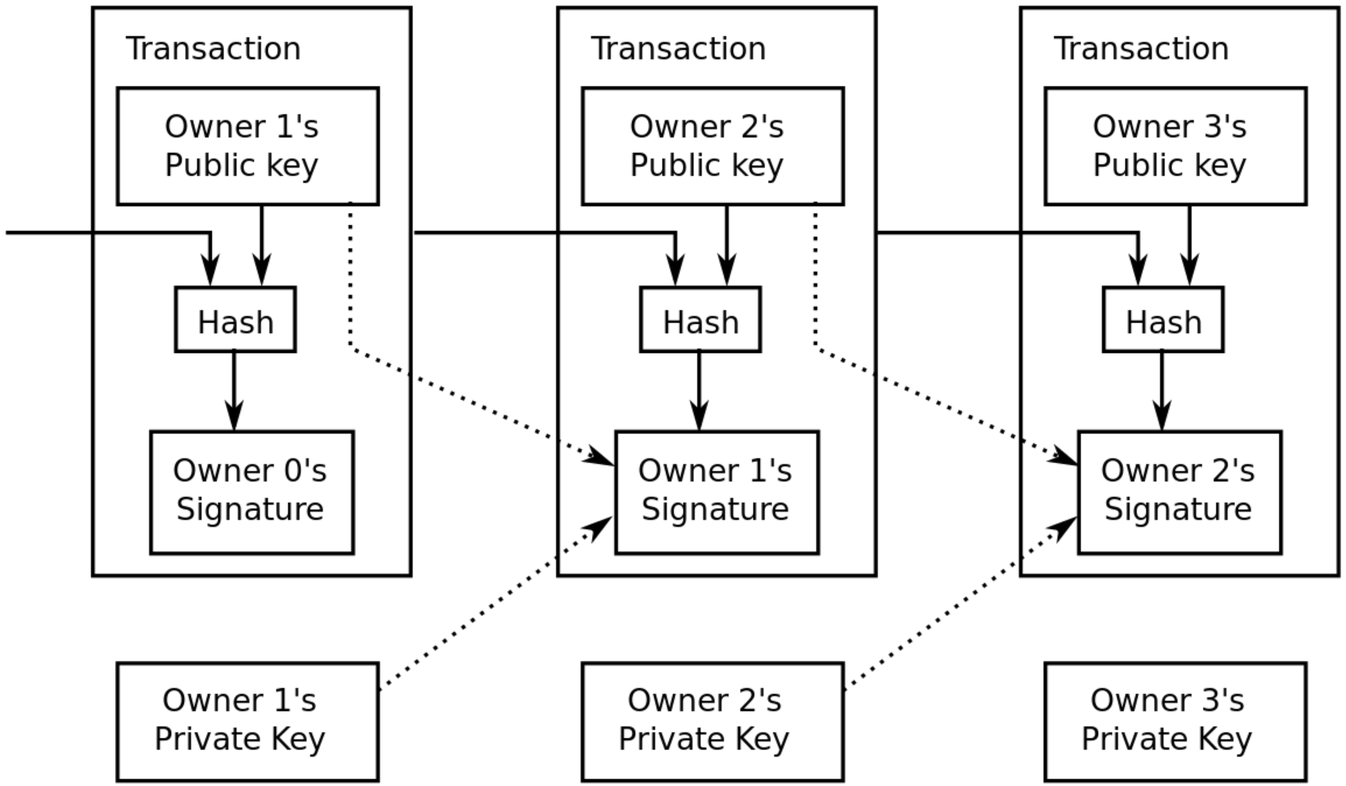
Mercury layer

A private non-custodial layer-2 for Bitcoin

Tom Trevethan - CTO Commerceblock

Bitcoin transactions



Transactions change ownership

Require confirmation in blockchain

Limited space: high fees and long confirmation times

Private keys …

Can we just send private keys?

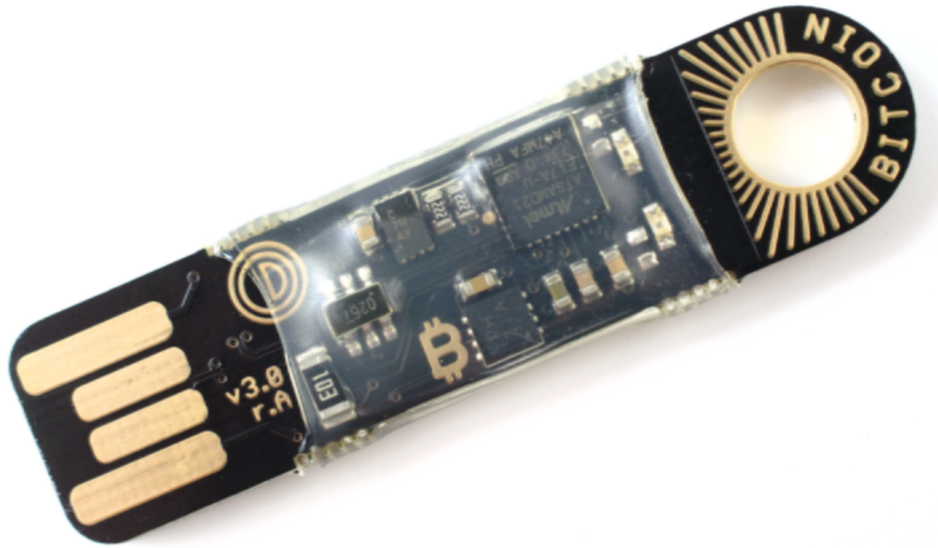
Off-chain: free, instant and private

BUT: have to trust sender to delete their key

Private keys …

How to prevent the previous owner from stealing?

1. Via trusted hardware

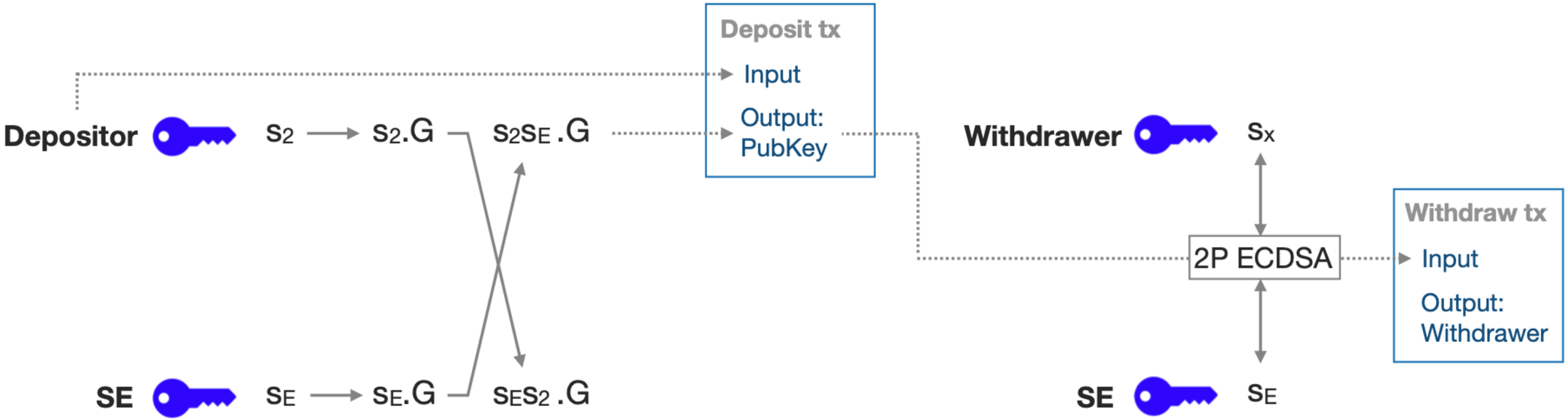


e.g. **Opendime**

Private keys …

How to prevent the previous owner from stealing?

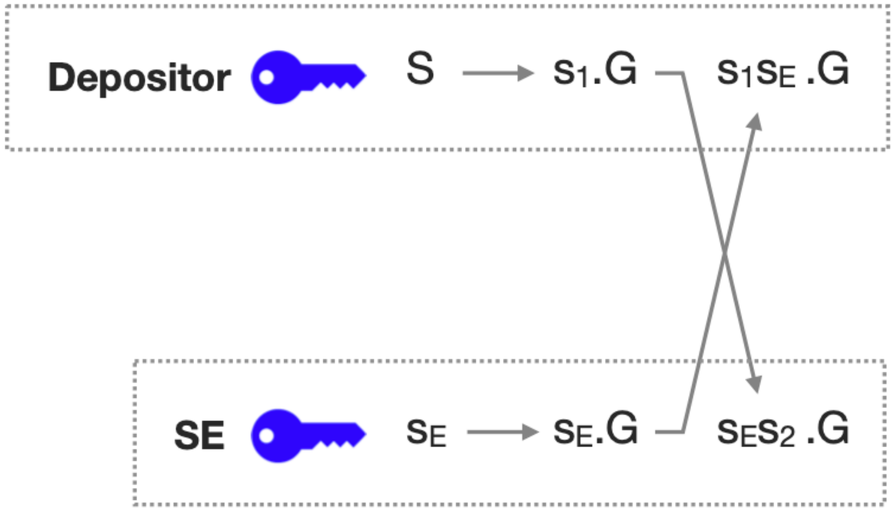
2. Using a trusted *key-share update server:* **Statechains**



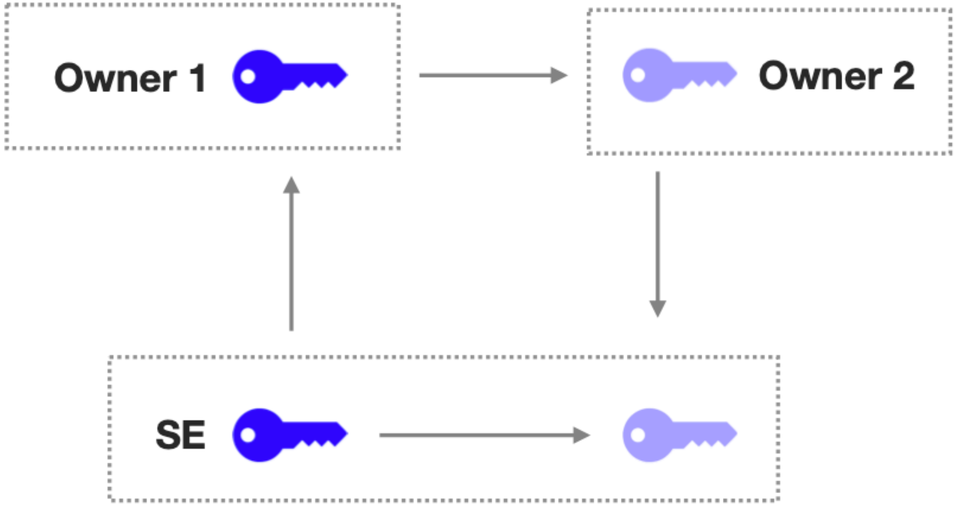
Mercury protocol

Single shared public key → P2(W)PKH on-chain output

Multiplicative private shares: $\mathbf{P} = s_1 s_E . G.$ (shared secret key $\mathbf{s_1 s_E}$ is never computed or known)



Transfer: Key share update protocol

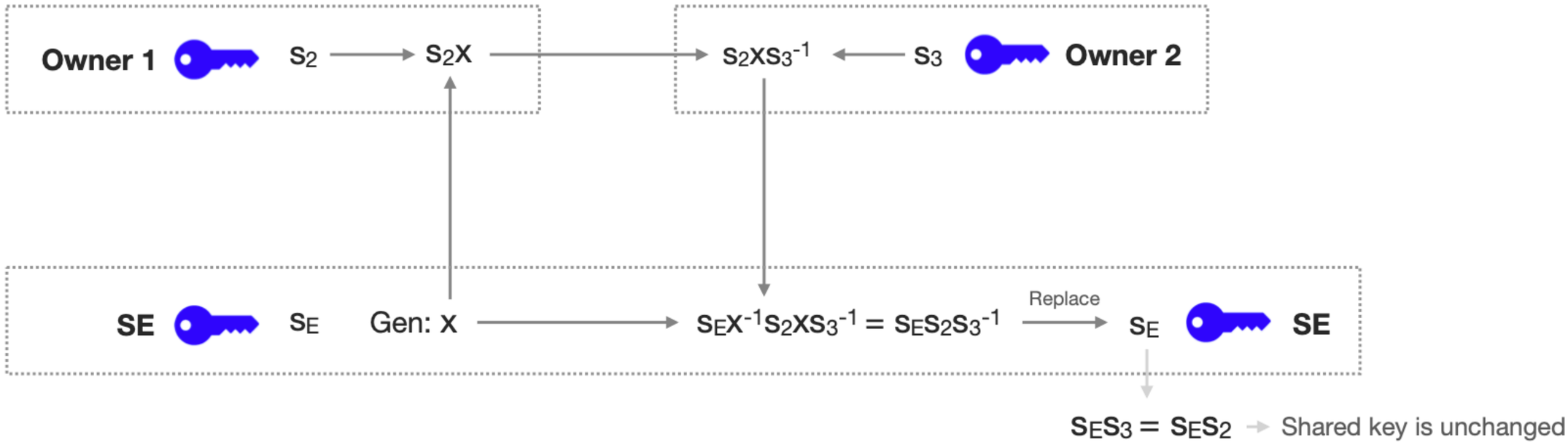


If the previous value of s_E is deleted, previous owners cannot steal the UTXO even if SE colludes or is hacked.

Mercury protocol

Transfer: Key share update protocol

Multiplicative private shares: **PubKey** = $s_1 s_1 .G$. (shared secret key $\mathbf{s_1 s_1}$ is never computed or known)



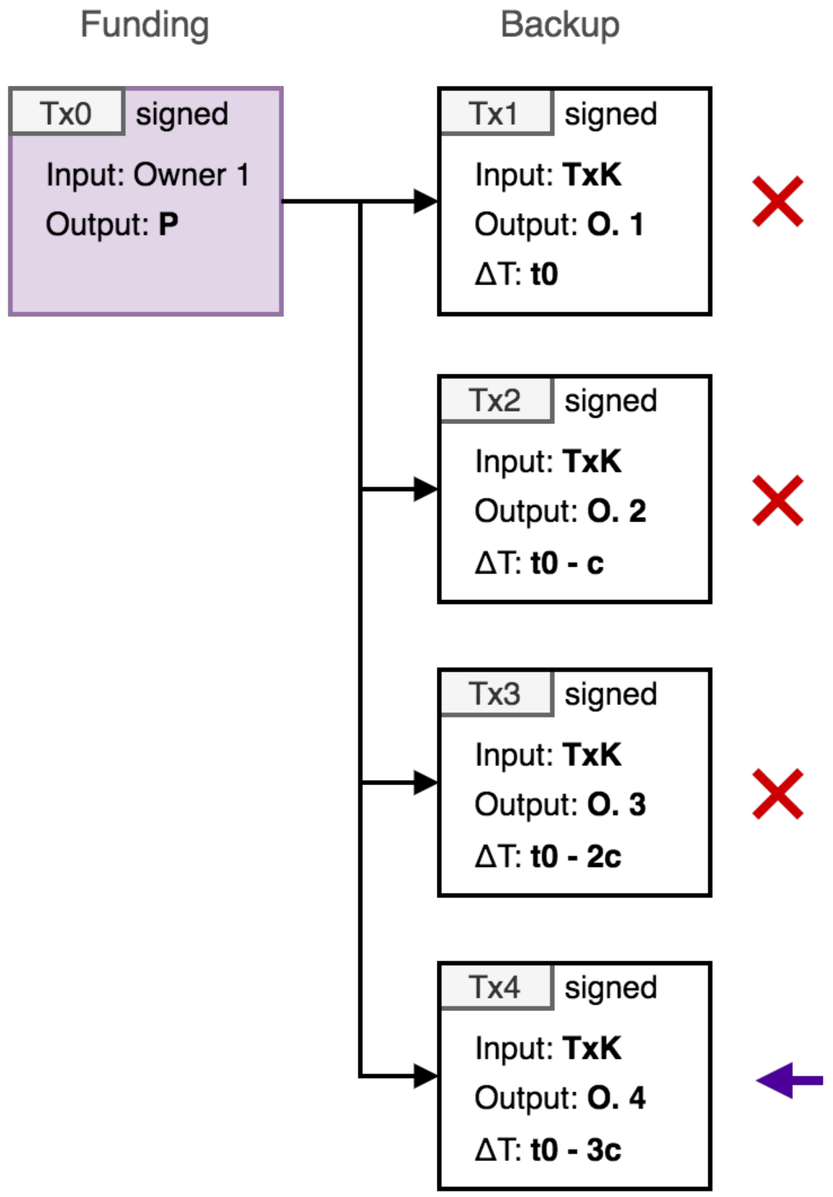
→ *Proactively* non-custodial

Mercury protocol

Off-chain (backup) transactions via
decrementing nLocktime

- Compatible with current Bitcoin protocol rules
- Previous owner attacks not possible

… this limits number of transfers and statecoin lifetime



Mercury wallet

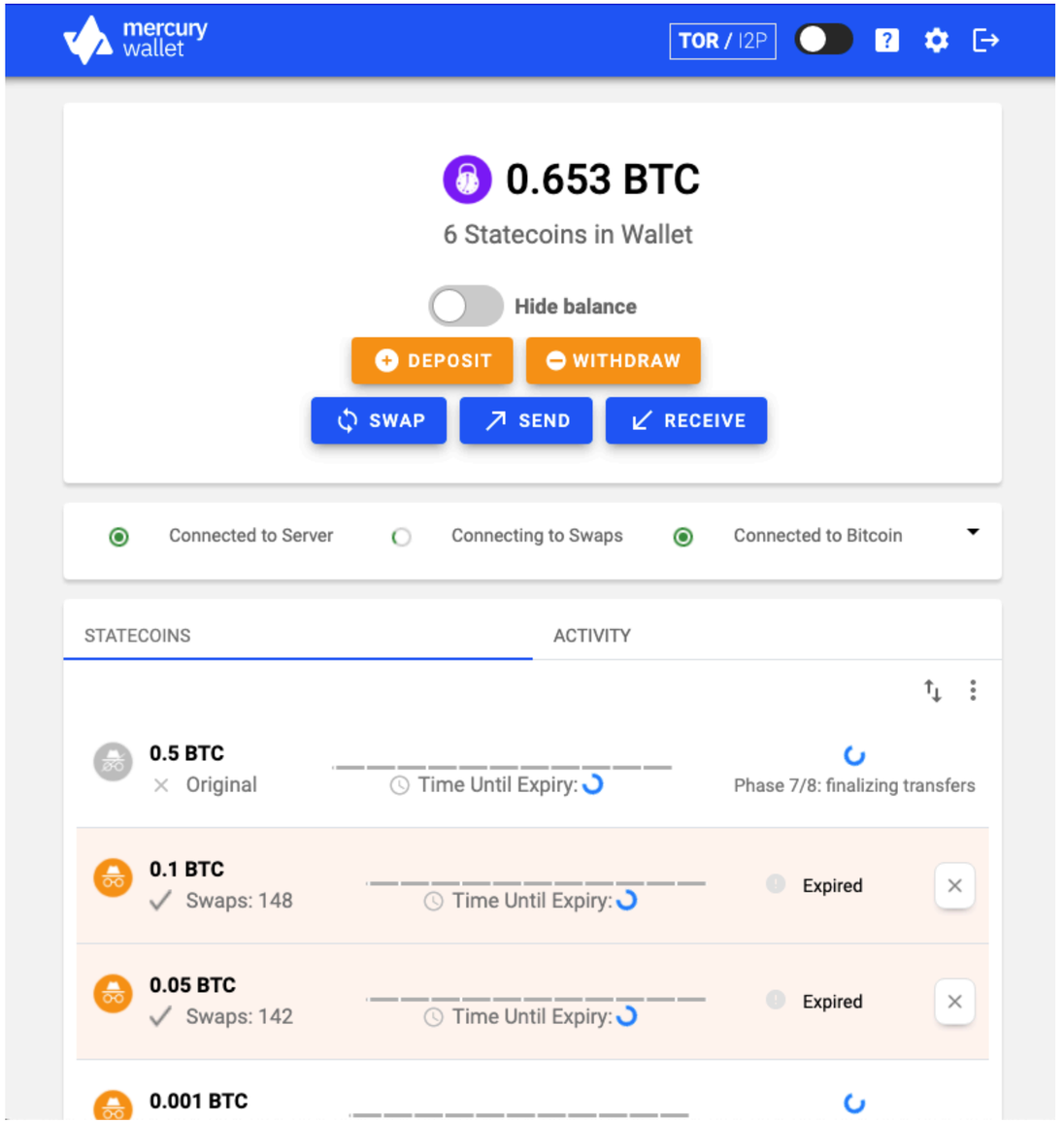
Launched 2021:

Over 25 BTC deposited

Over 80,000 coin swaps

But:

- Public keys and TxIDs known to server
- Statecoin UTXOs identifiable on-chain
- 2-Party ECDSA complex and slow



Mercury wallet

Launched 2021:

Over 25 BTC deposited

Over 80,000 coin swaps

But:

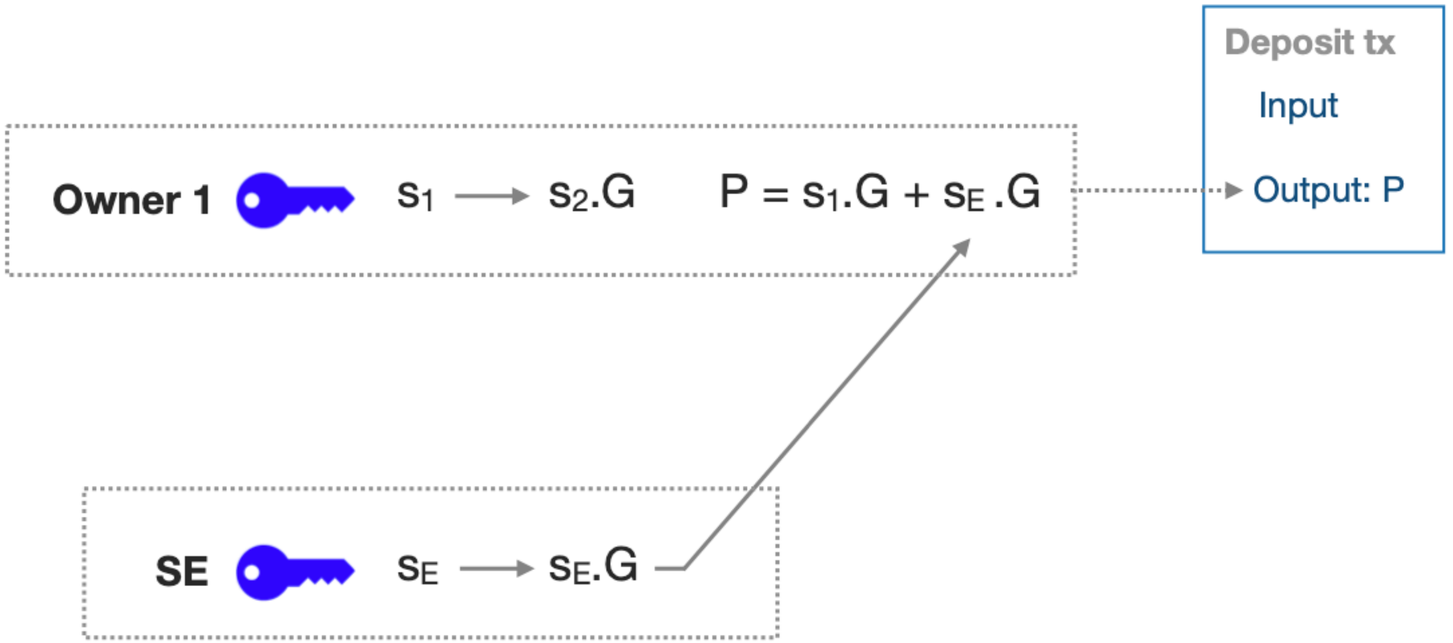
- Public keys and TxIDs known to server
- Statecoin UTXOs identifiable on-chain
- 2-Party ECDSA complex and slow

Solutions:

- Blind 2 party Schnorr signatures (MuSig)
- Statechain entity signature count
- Full client-side verification
  Server only reports *number* of co-signings
- Server completely blind to any on-chain identification

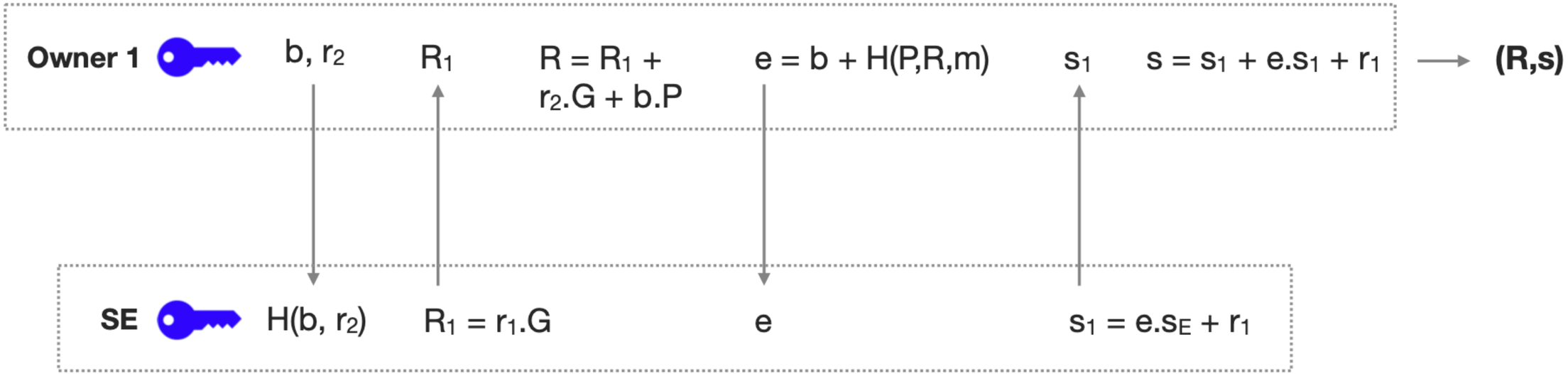
Blind 2-Party Schnorr

- Two parties required to generate a signature on shared public key
- One party (**SE**) does not learn: 1) The full shared public key. 2) The message (sighash) or 2) The final signature

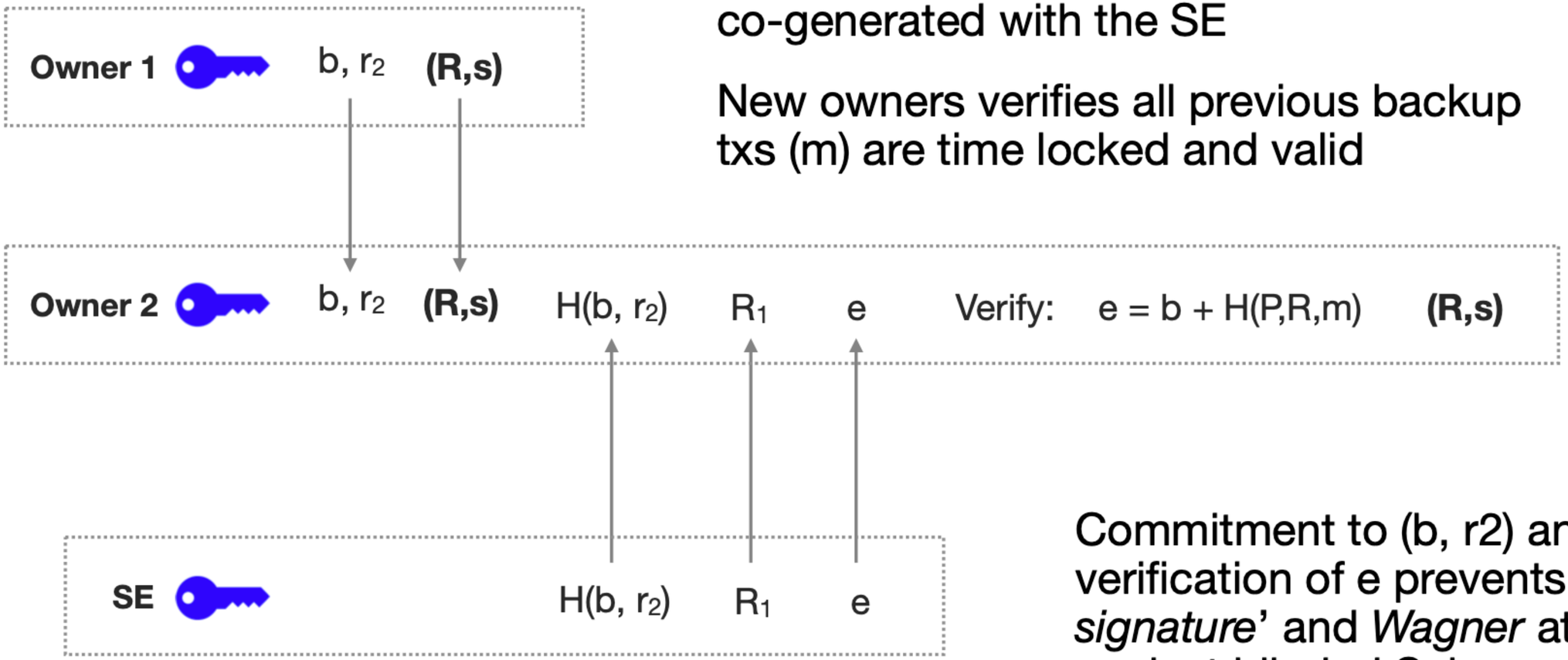


Blind 2-Party Schnorr

- Two parties required to generate a signature on shared public key
- One party (**SE**) does not learn: 1) The full shared public key. 2) The message (sighash) or 2) The final signature



Blind 2-Party Schnorr



New owner verifies all previous signatures co-generated with the SE

New owners verifies all previous backup txs (m) are time locked and valid

Commitment to (b, r2) and verification of e prevents '*one-more-signature*' and *Wagner* attacks against blinded Schnorr

Mercury Layer

Blind 2-party Schnorr enables a completely blind SE. SE trusted to report number of signatures, instead of enforcing rules.

SE is unable to have any knowledge of the on-chain identity of coins

Mercury Layer + atomic coinswaps completely on-chain transaction graph

Lightning latch protocol enables atomic statecoin/LN transactions

Blinded version of MuSig2 (with TC/HSM support)
Mercury layer server (Rust)
Mercury layer WASM (Rust) client
React-app client

github.com/commerceblock/mercurylayer
Telegram: mercurywallet

Bitcoin 2nd layers:

Lightning:

Trustless (in principle)

Verifiable

Unilateral exit

Requires liquidity

Arbitrary Payments (dependent on channels)

Mercury:

Trust required

Verifiable

Unilateral exit

Pro-actively non-custodial

Whole UTXOs

Completely blind

Fedimint/Cashu:

Trust required

Non-verifiable

Fully custodial

Arbitrary Payments

Private payments

Deposit/withdrawal identifiable

Liquid:

Trust required

Verifiable

Fully custodial

Arbitrary Payments

Peg in/out identifiable